SSE | Tutorial 3

Task 1

1. Answer the following questions:
   - What is the difference between URI and URL?
   - What is the meaning of the URL scheme?
   - What, why and how should be encoded in URLs?
2. Implement a class for parsing and generation of HTTP URLs[1]. Use the given *Task1-Template.zip* as a start point:
   - The constructor `Url(string urlStr)` should split the urlStr into URL components using a **regular expression** and fill in the instance variables `Scheme`, `Host` etc. One should assume the URL is correctly encoded.
   - The function `string ToString()` should concat the instance variables to the string representation of the URL.
   - The static method `string Encode(string s)` should convert all characters from `s`, which are not in `VALID_CHARACTERS` into the %-form and give the resulting string back.
   - The static method `string Decode(string s)` should convert all %-escaped characters from `s` and give the resulting string back.

Task 2

1. Explain the semantics of the following HTTP methods: GET, HEAD, PUT, DELETE, and POST. Which of them are **safe**, which are **idempotent**, and which are **cacheable**?
2. Explain the purpose of the following HTTP headers:
   - Host
   - Content-Type
   - Content-Length
   - Accept
   - User-Agent
   - Location

Task 3

Implement an HTTP message *parser* and *builder* based on the template *Task3-Template.zip* (take care of differentiation between request and response messages). Complete the methods *Parse* and *ToString*.

Task 4

1. What are the goals of HTTPS and how they are achieved?
2. What is the difference between HTTP and HTTPS request/response messages?

---

[1] For simplification purposes, it is enough that the given unit test passes (the solution should not be limited to the given URL though)

Homework

1. Inform yourself about the "chunked" transfer encoding and its purpose. Extend the HTTP message parser and builder from Task 3 with the support for “chunked” transfer encoding.
2. Based on the template *Homework-Template.zip* implement a server, which is able to **deliver** requested resources from the *HttpServer.DOCUMENT_ROOT* folder (for POST requests return only *201 Created*). Test your implementation in the browser.
3. Modify[2] the HTTP request implementation of Task 3 to request the following resource: https://www.tu-chemnitz.de (HTTPS)

[2] Use http://msdn.microsoft.com/en-us/library/system.net.security.sslstream.aspx as a reference